2006

A Template-Based Data Specification Framework For Modeling Physical Security Systems

Ashu Guru
University of Nebraska at Lincoln, ashuguru@gmail.com

Paul Savory
University of Nebraska at Lincoln, psavory2@gmail.com

A. Guru and P. Savory (2006) "A Template-Based Data Specification Framework For Modeling Physical Security Systems," ***Computers & Industrial Engineering***, Volume 50, Issue 3, pp. 195-201.

A Template-Based Data Specification Framework For Modeling Physical Security Systems

ASHU GURU
Department of Industrial and Management Systems Engineering
University of Nebraska – Lincoln

PAUL SAVORY
Department of Industrial and Management Systems Engineering
University of Nebraska – Lincoln

ALL CORRESPONDENCE SHOULD BE ADDRESSED TO:

Mailing: Dr. Paul Savory
175 Nebraska Hall
Industrial Engineering
University of Nebraska
Lincoln, NE 68588-0518

Phone: (402) 472-0657
Fax: (402) 472-1384

E-Mail: psavory@unl.edu

A. Guru and P. Savory (2006) "A Template-Based Data Specification Framework For Modeling Physical Security Systems," ***Computers & Industrial Engineering***, Volume 50, Issue 3, pp. 195-201.

A Template-Based Data Specification Framework For Modeling Physical Security Systems

ABSTRACT

Simulation studies often fail to provide any useful results due to its success being highly dependent on the skills of the analyst to understand a system and then correctly identify all the required data parameters and dependent variables. This paper describes a template-based framework to help identify and specify the components and data parameters for developing models of physical security systems. The layered framework consists of fifteen templates built on top of fourteen data primitives representing 119 data parameters. The modeling framework has been programmed as an internet-based web application and is simulation language independent. The usefulness of the framework was tested and shown to have a significant impact on improving the identification of system components and their associated data parameters.

Keywords: discrete-event simulation; physical security systems; templates; modeling framework

1. Introduction

Although simulation is one of the most innovative and cost-effective tools for system modeling and analysis, simulation studies often fail to provide any useful results (Annino & Russell, 1979; Keller et al. 1991; Robinson & Pidd, 1998). One reason is attributed to the fact that *model formulation* – a key step in a simulation study – requires an analyst to work from a sense of the problem, envision and assemble the key elements, and identify dependencies and relationships that logically comprise the variables of the actual system. Thus, the success of a simulation study is highly dependant on an analyst's domain knowledge, capability to understand the system components, their input parameters, and the interrelationships among those variables and

parameters. One approach for improving the efficiency, productivity, and quality of a model is to provide an analyst with a framework for helping identify the components of a system and their input parameters.

Development of such a framework is even more important when there is a large number of similar simulation studies being conducted within a single domain. One such domain is physical security systems. Since the terrorist attacks of September 11, 2001, the requirements of constantly managing and re-evaluating all direct and indirect risks in physical security systems has increasingly become more important. Numerous studies (Gatersleben & Weij, 1999; Jordan et al. 1998; Joustra & Dijk, 2001; Kyle 1998; Leone, 2002; Parizi & Braaksma, 1995; Saffarazadeh & Braaksma, 2000; Smith et al. 1999) have focused on this domain area.

Various methodologies, tools, and techniques have been proposed to assist a simulation analyst. In the field of automated model development, generic or template-based simulation modeling approaches have been proposed as a solution for domain specific modeling assistance. This approach consists of using an available set of pre-built, ready to use modeling objects, modules, or models of common simulation situations. Using these modules, an analyst simply "switches on" or "switches off" the model parameters of the generic module to fit to the system under study. Much previous work (Brown & Powers, 2002; Diaz-Calderon et al. 2000; Kasputis & Ng, 2000; Mackulak et al. 1998; Mackulak & Cochran, 1990; Page & Opper, 1999; Son et al. 2000) explores using generic models. Similarly, Steele et al. (2002) and Overstreet & Nance (2003) have looked at designing and reusing simulation components between simulation studies. Unfortunately, much of this previous research is focused on developing an "executable" simulation model. That is, they are specific to a simulation language, software program, and/or simulation package. For instance, several simulation software programs provide pre-defined

software templates or tools to guide a user in building a model of a telephone call center. In comparison, this research does not seek to implement software specific simulation components, but rather is focused on developing a general framework that will assist a simulation analyst in identifying the significant input modeling parameters important in creating the model of a physical security system. By presenting these research results in the open literature will make them available to all developers and researchers as a first step for creating standard, non-proprietary templates and specifications that are compatible across simulation packages..

The remaining sections of this paper reveal the development of the model framework and shows how it can impact the process of defining and collecting the data for a simulation study. Section 2 defines the data primitives – the building blocks of the framework – and their parameter specification. Section 3 combines the data primitives into logical templates for common security system implementations. Section 4 highlights a web application that implements the framework. Section 5 shares details on how the framework was validated. Section 6 concludes with an overview of the research.

2. Data Primitives

The basic building block for the physical security formalism is a set of data primitives. Each assists an analyst in identifying the key system parameters for which the data may need to be collected from the real system. Based on an extensive review of the modeling structure and components from simulation environments (e.g., SIMAN, ARENA, EXTEND, SIMUL8, PROMODEL), three classes of data primitives are defined: Entity, Model, and Experimental. Table 1 presents a listing of all the data primitives defined for each class.

<<< Insert Table 1 approximately here >>>

Table 2 shows the definition table for the queue primitive. Table 3 is the definition table for a transporter device. Similar tables have been developed for the other 12 data primitives (Guru, 2004). Each primitive definition table has four columns. The first column is for the Parameter Name and hence contains the name of the configurable parameter for the primitive being defined. The second column of the table defines the Parameter Type. Parameter type can have the following values:

- *Native* – the corresponding parameter is native to the defined primitive. For example the *Work Center* primitive type has a parameter titled: *Name.* This parameter is of native type because it is a string value that defines this primitive's instance
- *Reference* - the corresponding parameter is a reference to another data primitive. For example the *Work Center* primitive has a parameter *Resources*. This parameter is a reference parameter another data primitive, namely *Resources.*

The third column lists the type of value that can be assigned to the parameter. Basic types include: Integer, Boolean (i.e., Yes or No), Distribution (i.e., statistical probability distribution), and String (i.e., for character text string). In addition, there can be Arrays of these basic types. The final column in a primitive definition provides an explanation of the parameter.

<<< Insert Table 2 approximately here >>>

<<< Insert Table 3 approximately here >>>

3 Physical Security System Templates

Using the defined classes and data primitives, fourteen logical templates have been defined for representing common physical security sub-systems. The templates were formed by collecting, grouping, and relating the data primitives to represent common real world sub-systems. The

templates are classified into five security sub-system categories: (1) Inspection and Detection System, (2) Identity Management System, (3) Perimeter Protection and Intrusion Detection System, (4) Access Control System, and (5) Entity Handling System. A single template may fall under one or more categories. Table 4 depicts this classification. Each of the developed templates contain the data primitives relevant for building the conceptual model of the respective physical security system.

<<< Insert Table 4 approximately here >>>

As an example, consider an Explosive Detection (ED) machine. An ED machine is able to detect explosives and hazardous liquids as well as metallic weapons concealed in an object such as luggage. The technology needed to do this includes quadrupole resonance (closely related to magnetic resonance technology used in the medical industry) and magnetic sensors. Figure 1 shows the issues impacting an ED machine's operation. This research's developed ED template links together the necessary data primitives for modeling a typical ED system layout. Specially, there is a queue for storing luggage to be processed. The queue has a route-out selection criterion for when the queue is full or for when the luggage delay exceeds some maximum waiting time. The ED machine (the work center) has a route-in priority selection. In this instance, there is only one luggage queue from which it is pulling from. The ED machine can have a work schedule (hours available, not available), a failure description (mean time between breakdown and mean time to repair, etc.) and once processed, a route-out selection for where to send luggage. In addition, a worker (a resource) is required for operating the ED machine. The availability of the worker can be impacted by a schedule and failures. Table 5 lists the set of eight data primitives used to define the EDS template. Similar linkages of data primitives have been assembled for each of the other thirteen physical security system templates (Guru, 2004).

<<< Insert Figure 1 approximately here >>>

<<< Insert Table 5 approximately here >>>

4. Computer Implementation

A web-based application has been developed to assist an analyst to use the formalism. The application consists of a MySQL database engine and an Apache Web server using Perl CGI scripting. Using the application, a user will select all the templates and additional primitives for modeling the physical security system under study. Once complete, the application will generate the data primitive tables for collecting all of the key details and information. Figure 2 is a screen-shot of the application.

<<< Insert Figure 2 approximately here >>>

5. Results of Validation Testing

An experiment was designed to evaluate the usefulness and effectiveness of the developed framework. Test subjects were forty-five graduate and undergraduate Industrial Engineering students – all potential future users of the framework and the application. Each test subject was provided with a description of one of three hypothetical modeling scenarios: Airport Screening, Smart Parking facility, and Mail Purification facility. Each was asked to analyze the given scenario and identify all the input parameters (*i.e.,* parameters for which data should be collected from the system under study) that he/she perceived to be necessary in performing a system analysis using computer simulation. The input parameters identified by the test subjects were compared to those generated by the web-based applications. For example, for the Airport Screening scenario, test subjects identified an average of 15 data parameters. In comparison, the

web application generates a listing of 240 possible data parameters to be collected. Overall, it was found that test subjects identified only 18% of the data parameters that potentially are important to be incorporated in the models of the three scenarios. Additionally, in comparing results for each scenario, it was observed that with increased complexity of the description, the percentage of missed parameters increases, and that the average number of parameters identified by a test subject is highly dependent on their experience and skills. Such results support the belief that the developed framework is an important first step in providing a methodology that will make simulation less *modeler dependent* and help improve the consistency of model quality.

6. Conclusions

Simulation is a widely used tool for modeling and analyzing real systems for answering capacity and feasibility questions. Unfortunately, simulation studies often fail to provide any useful results because it is highly dependent on the skills of the analyst to understand the system's problem and then correctly identify the required modeling parameters and dependent variables. As a partial solution, this research presents a language-independent framework that aids an analyst in identifying all the possible components and data parameters necessary for building a conceptual model of common physical security systems. The resulting model framework focuses on identifying variables and data parameters that potentially should be collected when modeling a physical security system.

Since the framework is simulation-language independent, it offers the advantage that common or standard physical security templates can be developed. Such an approach would encourage transferability of models to different simulation software programs since they would be build using the same data primitive definitions. As such, simulation models could potentially

be saved in an open format that is accessible by different simulation packages. For example, the OpenDocument OASIS Standard (OASIS 2006) for word processing programs. Even if such an approach were not possible, given an underlying commonality in model definition, there is the potential that a converter program could be developed for transferring a simulation model from one software format to another. For instance, converter applications are commonly used by word processing programs (Microsoft Word, WordPerfect, Open Office) to open user files saved in various formats.

Although this research focuses on physical security systems as the domain for identifying the components and parameters, the results highlight the applicability of developing standard specifications and template-based frameworks for other domains.

7. References

Annino J.S., & E.C. Russell. (1979). The Ten Most Frequent Causes of Simulation Analysis Failure and How to Avoid Them. Simulation, 60,137-140.

Brown, N. & S. Powers. (2000). Simulation in a Box: (A Generic Reusable Maintenance Model). In: J.A. Joines, R.R. Barton, K.Kang & P.A. Fishwick, Proceedings of the 2000 Winter Simulation Conference (pp. 1050-1056).

Diaz-Calderon, A., C. Paredis, & P. Khosla. (2000). Organization and Selection of Reconfigurable Models. In: J.A. Joines, R.R. Barton, K.Kang & P.A. Fishwick, Proceedings of the 2000 Winter Simulation Conference (pp. 389-393).

Gatersleben, M.R. and S.W Weij. (1999). Analysis and Simulation of Passenger Flows in an Airport Terminal. In: P.A. Farrington, H.B. Nembhard, D.T. Sturrock & G.W. Evans, Proceedings of the 1999 Winter Simulation Conference (pp. 1226-1231).

Guru, A. (2004). A template-based approach for simulation model specification of physical security systems. PhD Dissertation. Lincoln, NE: University of Nebraska

Jordan, S.E., M.K. Snell, M.M. Madsen, J.S. Smith & B.A. Peters. (1998). Discrete-Event Simulation for the Design and Evaluation of Physical Protection Systems. In: D.J. Medeiros, E.F. Watson, J.S. Carson & M.S. Manivanna, Proceedings of the 1998 Winter Simulation Conference. (pp. 899-904).

Joustra, P.E. & N.M. V. Dijk. (2001). Simulation of Check-In at Airports. In: B.A. Peters, J.S. Smith, D.J. Medeiros & M.W. Rohrer, Proceedings of the 2001 Winter Simulation Conference. (pp. 1023-1028).

Kasputis, S. & H.C. Ng. (2000). Composable Simulations. In: J.A. Joines, R.R. Barton, K.Kang & P.A. Fishwick, Proceedings of the 2000 Winter Simulation Conference (pp. 1577-1584).

Keller L., C. Harrell & J. Leavy. (1991). The Three Reasons Why Simulation Fails. Industrial Engineering. 23(4), 27-31.

Kyle, R.G. (1998). Washington Dulles International Airport Passenger Conveyance Study. In: D.J. Medeiros, E.F. Watson, J.S. Carson & M.S. Manivanna, Proceedings of the 1998 Winter Simulation Conference. (pp. 1119-1123).

Leone, K. (2002). Security System Throughput Modeling. Atlantic City: Transportation Security Administration – Aviation Security Research and Development Division.

Mackulak, G.T. & J.K. Cochran. (1990). The Generic Specific Modeling Approach: An Application of Artificial Intelligence to Simulation. In: IIE Integrated Systems Conference & Society of Integrated Manufacturing Conference Proceedings. (pp. 82-87).

Mackulak, G.T., F.P. Lawrence & T. Colvin. (1998). Effective Simulation Model Reuse: A case study for AMHS modeling. In: D.J. Medeiros, E.F. Watson, J.S. Carson & M.S. Manivanna, Proceedings of the 1998 Winter Simulation Conference. (pp. 979-984).

OASIS (Organization for the Advancement of Structured Information Standards). (2006) http://www.oasis-open.org/home/index.php visited May 2006.

Overstreet, C.M. & R.E. Nance. (2003). "Issues in Enhancing Model Reuse" http://www.thesimguy.com/GC/papers/WMC02/G108_OVERSTREET.pdf. visited December 2005.

Page, E.H. & J.M Opper. (1999). Observation on the Complexity of the Composable Simulation. In: P.A. Farrington, H.B. Nembhard, D.T. Sturrock & G.W. Evans, Proceedings of the 1999 Winter Simulation Conference (pp. 553-560).

Parizi, M.S. & J.P. Braaksma. (1995). An optimum Resource Utilization Plan for Airport Passenger Terminal Building. Washington, D.C: Transportation Research Record (1506) – National Research Council (pp. 34-43).

Robinson, S. & M. Pidd. (1998). Provider and Customer Expectations of Successful Simulation Projects. Journal of Operational Research Society 49: 200-209.

Saffarzadeh M. & J.P. Braaksma. (2000). Optimum Design and Operation of Airport Passenger Terminal Building. Washington, D.C: Transportation Research Record (1703) – National Research Council (pp. 72082).

Smith, J.S., B.A. Peters, S.E. Jordan & M.K. Snell. (1999). Distributed Real-Time Simulation for Intruder Detection System Analysis. In: P.A. Farrington, H.B. Nembhard, D.T. Sturrock & G.W. Evans, Proceedings of the 1999 Winter Simulation Conference (pp. 1168-1173).

Son, Y.J., A.T. Jones & R.A. Wysk. (2000). Automatic Generation of Simulation Models from Neutral Libraries: An Example. In: J.A. Joines, R.R. Barton, K.Kang & P.A. Fishwick, Proceedings of the 2000 Winter Simulation Conference (pp. 1558-1567).

Steele M.J., M.M.G. Rabadi & G. Cates. (2002). Generic Simulation Models of Reusable Launch Vehicles. In J. L. Snowdon, J. M. Charnes, E. Yucesan, C-H Chen, Proceedings of the 2002 Winter Simulation Conference (pp. 747-753).

Table 1. Classes and Associated Data Primitives

Class	Primitive	Description
Entity	Entity	Represent the physical or conceptual entities(s) that flow through a security system.
Model	Entry Point	Place where an entity appears in the simulation model for the first time.
	Queue	Place where an entity waits until a resources or work centers is available to process it.
	Work Center	Where an entity is processed and released to continue in the model.
	Exit Point	Where a work item that is complete (or otherwise finished) leaves the system.
Experimental	Clock	How time units increment for the simulation model.
	Route-In	Encapsulates the decision operations for a referencing primitive that has an option of selecting from more than one choice.
	Route-Out	Encapsulates the destination routing decision operations for a referencing data primitive.
	Resources	Resources are the people and other objects associated with a work center in order for the work center to process the entity.
	Path	Path is the lane joining modeling primitives in a simulation model.
	Conveyor	Material handling device for transporting entities in a system.
	Transporter	Material handling device used to transfer entities in between work centers.
	Work Schedule	Define the time-dependent work schedule and the capacity of the referencing data primitives.
	Failure Schedule	Define the characteristics of failures and breakdowns that are associated with referencing primitives.

Table 2. Configuration parameters of queue data primitive

Parameter	Parameter Type	Value Type(s)	Description
Name	Native	String	Unique name of the queue.
Priority	Native	Integer	Defines the priority level that may be used by a route-in type variable of the work center
Capacity	Native	Distribution	Defines the maximum capacity of the queue or how many maximum entities can wait in this queue at one time
Minimum wait time	Native	Distribution	Minimum time the entity should wait in the queue
Queue order Rule	Native	FIFO	These are the guidelines for scheduling the entities and assigning them the order in which they wait until the work center and the resource are available FIFO- First in first out LIFO- Last in first out HVF- High value first based on a parameter LVF- Low value first based on a parameter
		LIFO	
		HVF (parameter)	
		LVF (parameter)	
Shelf life	Native	Distribution	This is the maximum time entity waits. If past, the entities will balk, or leave the queue
Balking route-out	Reference	Reference type	This is the routing logic of what happens to the entities when they leave the queue either because they expired the shelf life or because the queue was at its capacity
Start up population	Native	Distribution	This is the number of entities that are waiting in the queue at the beginning of the simulation run

Table 3. Configuration parameters of transporter data primitive

Attributes	Parameter Type	Value Type(s)	Description
Name	Native	String	Unique name of the transporter primitive.
Travel speed	Native	Distribution	This is the travel speed of this transporter
Acceleration	Native	Distribution	This attribute defines the acceleration with which transporter gains speed
Deceleration	Native	Distribution	This attribute defines the deceleration with which transporter looses speed
Home	Reference	Reference type	This is the definition of the home of the transporter. It is defined by the reference to one work center
Return home if idle	Native	Boolean	This defines if the transporter should return home when it is idle
Route-out	Reference	Reference type	Route-out type associated with this transporter
Work Schedule	Reference	Reference	Work schedule associated with this transporter. This parameter also defines the capacity of the transporter (i.e., number of entities that this transporter can transport simultaneously)
Failures Schedule	Reference	Reference	Failure schedule associated with this transporter
Loading time	Native	Distribution	The distribution that defines the loading time for this transporter
Unloading time	Native	Distribution	The distribution that defines the unloading time for this transporter
Path(s)	Reference	Array of References	Path(s) associated with this transporter
Length	Native	Distribution	Length of transporter used to find out possibilities of jam
Initial Position	Reference	Reference type	Initial Position of the transporter. Defined by reference to the work center

Table 4. Classifications and development of templates of physical security systems

Inspection and Detection System	**Identity Management System**	**Perimeter Protection and Intrusion Detection System**	**Access Control System**	**Entity Handling System**
Explosive Detection Machine (including X-ray Inspection, Mail Room X-ray Inspection Machine)	Automatic Vehicle Identification (AVI) Machine	Communications Transceivers	Automatic Vehicle Identification (AVI) Machine	Laser Measurement Equipment
Handheld Metal Detector	Biometric or Touchpad Access Control Device	Entrance Door (Slide, Swing and Rotation and Turnstiles)	Biometric or Touchpad Access Control Device	
K-9 Unit	Card/Ticket Reader Machine		Card/Ticket Reader Machine	
Mail Purification Equipment	License Plate Recognition (LPR) Machine		Entrance Door (Slide, Swing and Rotation and Turnstiles)	
Mobile X-ray Inspection Machine	Token Dispenser Machine			
Walk-through Metal Detector				

Table 5. Data Primitives used to define the EDS template

Primitive Type	Class	Explanation
Entity	Entity	Representing real world luggage
Queue	Model	Waiting place for the luggage arriving at the EDS
Work Center	Model	The EDS machine for processing luggage
Route-in	Experimental	Defines how luggage is assigned to the EDS
Route-out	Experimental	(2 off) Defines where luggage goes from the waiting queues and then after processing at the EDS
Resources	Experimental	Representative of workers or operator for the EDS
Work schedule	Experimental	(2 off) Configure the work schedule of the EDS and operator
Failures schedule	Experimental	(2 off) Configure the failures and breakdowns for the EDS and operator

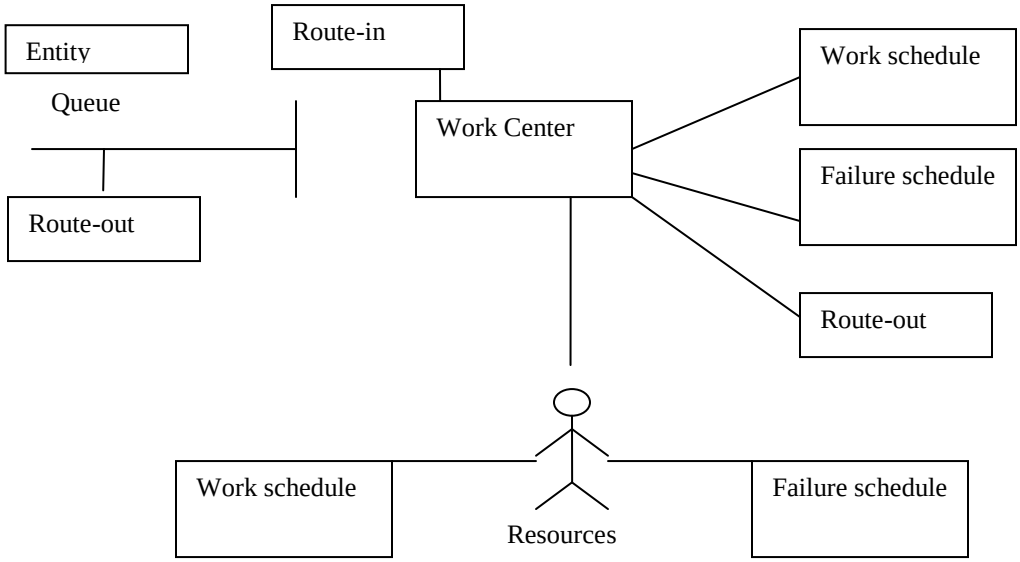

Figure 1. Components of an explosive detection machine template

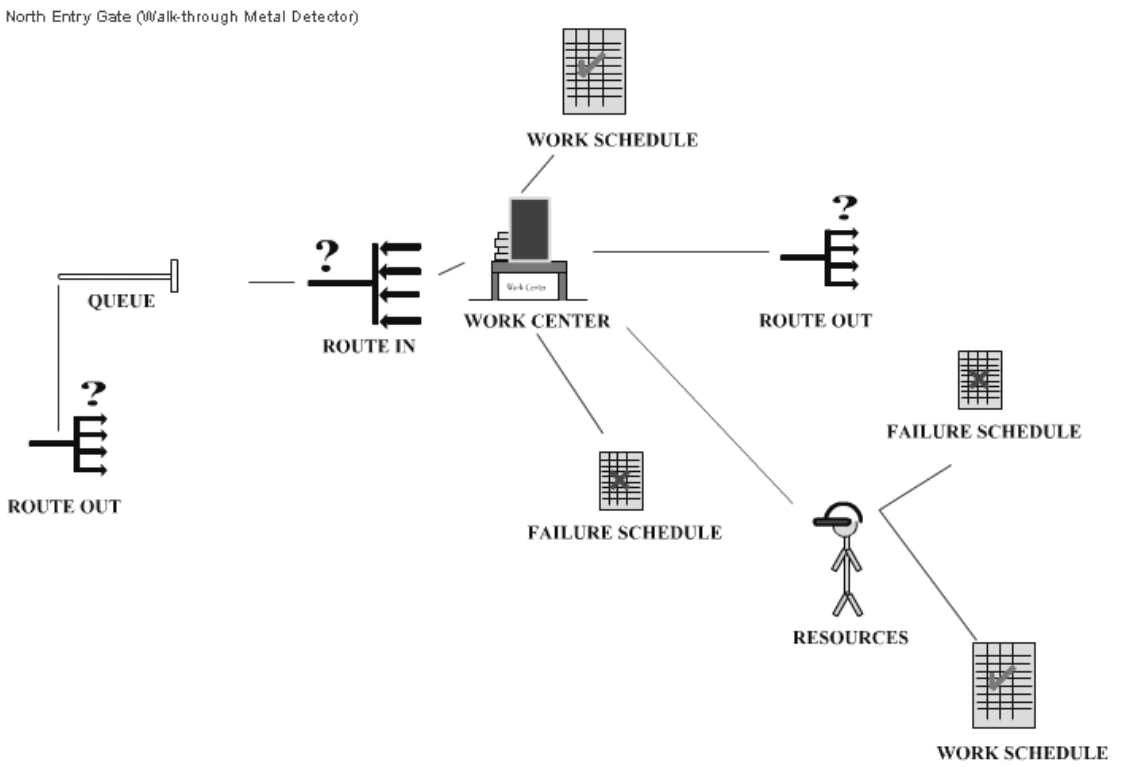

Figure 2. Screen shot of web application for creating data specification tables